

# PSTricks

---

## News - 2010

**new macros and bugfixes for the basic package pstricks**

November 3, 2010

2010

Package author(s):  
**Herbert Voß**

## Contents

<b>I. pstricks – package</b>	<b>3</b>
<b>1. General</b>	<b>3</b>
<b>2. pstricks.sty</b>	<b>3</b>
2.1. New optional argument . . . . .	3
<b>3. pstricks.tex (2.13– 2010/11/01)</b>	<b>3</b>
3.1. Coordinates . . . . .	3
3.2. New optional arguments . . . . .	4
3.3. Macro \psellipse . . . . .	8
3.4. Macro \psellipticarc . . . . .	8
3.5. Option algebraic . . . . .	10
<b>4. New linestyle symbol</b>	<b>12</b>
<b>5. Numeric functions</b>	<b>17</b>
<b>6. Numeric functions</b>	<b>17</b>
6.1. \pstFPadd, \pstFPsub, \pstFPMul, and \pstFPdiv . . . . .	17
6.2. \pstFPMul and \pstFPDiv . . . . .	18
<b>7. The PostScript header files</b>	<b>18</b>
7.1. pstricks.pro . . . . .	18
7.2. pst-algparser.pro . . . . .	19
7.3. IfTE . . . . .	20
7.4. Derive function . . . . .	22
7.5. Special arrow option arrowLW . . . . .	22
<b>8. \psforeach and \psForeach</b>	<b>24</b>
<b>II. pst-node – package</b>	<b>25</b>
<b>9. pst-node.tex</b>	<b>25</b>
<b>III. pst-plot – package</b>	<b>25</b>
<b>10pst-plot.tex</b>	<b>25</b>
<b>11List of all optional arguments for pstricks</b>	<b>26</b>
<b>References</b>	<b>26</b>

# Part I.

## pstricks – package

### 1. General

There exists a new document class `pst-doc` for writing PSTricks documentations, like this news document. It depends on the KOMA-Script document class `scrartcl`. `pst-doc` defines a lot of special macros to create a good index. Take one of the already existing package documentation and look into the source file. Then it will be easy to understand, how all these macros have to be used.

When running `pdflatex` the title page is created with boxes and inserted with the macro `\AddToShipoutPicture` from the package `eso-pic`. It inserts the background title page image `pst-doc-pdf` to use directly `pdflatex`. When running `latex` the title page is created with PSTricks macros. This allows to use the Perl script `pst2pdf` or the package `pst-pdf` or `auto-pst-pdf` or any other program/package which supports PostScript code in the document.

### 2. pstricks.sty

#### 2.1. New optional argument

With the setting of the optional argument `pdf` the package `auto-pst-pdf` will be loaded by PSTricks. This requires that you run `pdflatex` as

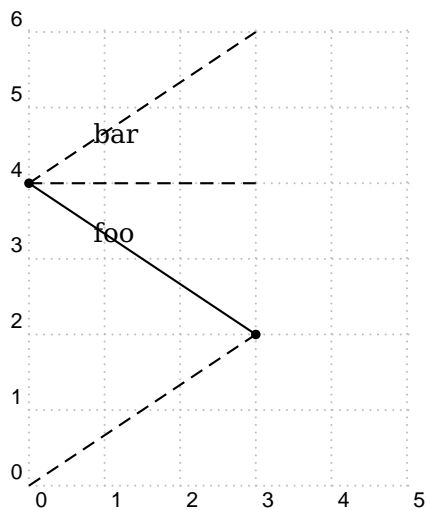
<pre>pdflatex --shell-escape &lt;file&gt; % T<sub>E</sub>X Live users pdflatex --enable-write18 &lt;file&gt;% MiK<sub>T</sub>E<sub>X</sub> users</pre>
--

The package exports the `pspicture` environments into single images which are collected in a created file `<file>-pics.pdf` and inserted automatically in the last `pdflatex` run.

### 3. pstricks.tex (2.13– 2010/11/01)

#### 3.1. Coordinates

With the setting `\SpecialCoor` the package allows different kinds of coordinates. The macro `\uput` can now be used in a different way. The default behaviour for nodes with a relative point puts its argument *without* rotation on the line  $\overline{AB}$ . When using the prefix `>` before the node or the  $x$ -value for cartesian coordinates, the behaviour is different. Now the angle between the line  $\overline{OB}$  and the horizontal line is taken into account and the placement of the argument of `\uput` is different to the default behaviour.



```

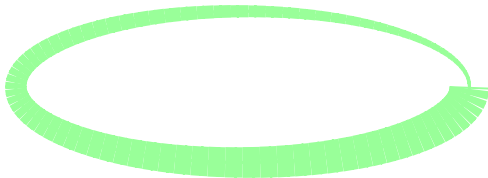
1 \begin{pspicture}[showgrid](5,6)
2 \pnode(0,4){A}\psdot(A)
3 \pnode(3,2){B}\psdot(B)
4 \psline(A)(B)
5 \psline[linestyle=dashed](A)(3,4)
6 \psline[linestyle=dashed](A)(3,6)
7 \psline[linestyle=dashed](B)
8 \uput{1cm}[(B)](A){foo}
9 \uput{1cm}[(B)](>A){bar}
10 \end{pspicture}

```

### 3.2. New optional arguments

The new arguments are only valid for the macros `\psellipse`, `\pscircle`, `\psarc`, `\psellipticarc`, `\pscurve`, `\psplot`, and `\psparametricplot`.

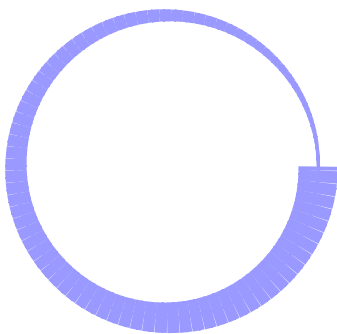
<i>name</i>	<i>type</i>	<i>default</i>	<i>description</i>
<code>startLW</code>	<i>length</i>	<code>\pslinewidth</code>	starting linewidth
<code>endLW</code>	<i>length</i>	<code>\pslinewidth</code>	ending linewidth
<code>startWL</code>	<i>integer</i>	380	starting wave length
<code>endWL</code>	<i>integer</i>	780	ending wave length
<code>variableLW</code>	<i>boolean</i>	false	use variable linewidth
<code>variableColor</code>	<i>boolean</i>	false	use variable color



```

1 \psset{endLW=15pt}
2 \begin{pspicture}(-3.5,-2.5)(3.5,2.5)
3 \psellipse[linejoin=2,variableLW,startLW=1pt,
4   linecolor=green!40](0,0)(3,1)
5 \end{pspicture}

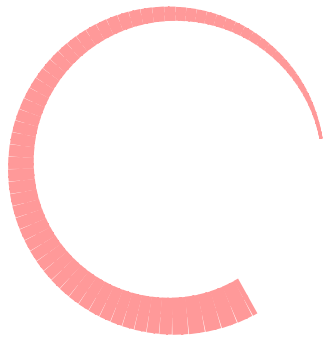
```



```

1 \psset{endLW=15pt}
2 \begin{pspicture}(-2.5,-2.5)(2.5,2.5)
3 \pscircle[variableLW,startLW=1pt,
4   linecolor=blue!40]{2}
5 \end{pspicture}

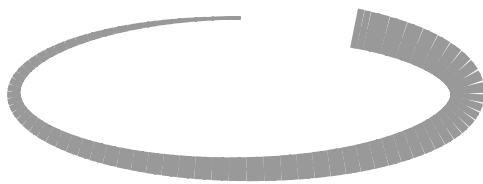
```



```

1 \psset{endLW=15pt}
2 \begin{pspicture}(-2.5,-2.5)(2.5,2.5)
3 \psarc[variableLW,startLW=1pt,
4   linecolor=red!40](0,0){2}{10}{300}
5 \end{pspicture}

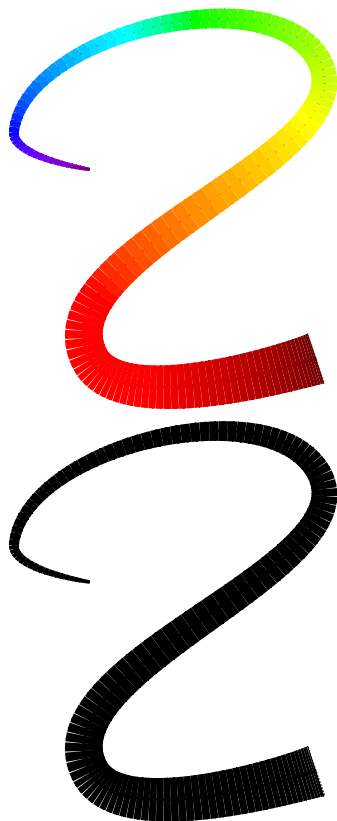
```



```

1 \psset{endLW=15pt}
2 \begin{pspicture}(-3.5,-2.5)(3.5,2.5)
3 \psellipticarc[variableLW,startLW=1pt,
4   linecolor=black!40](0,0)(3,1){90}{30}
5 \end{pspicture}

```



```

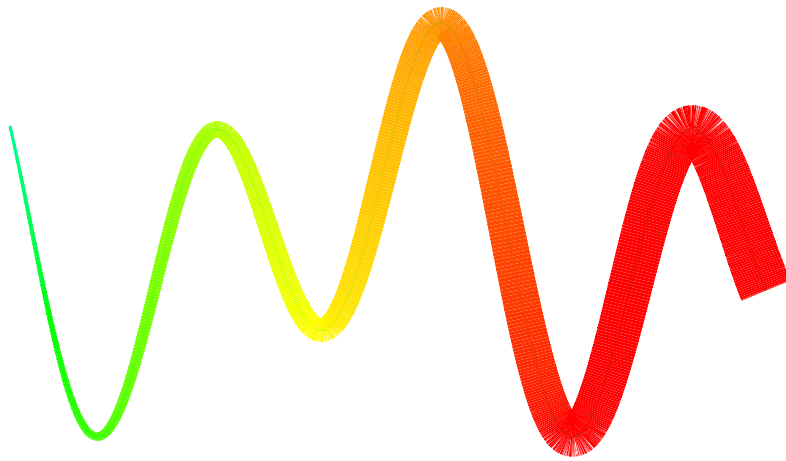
1 \begin{pspicture}(-2.5,-2.5)(2.5,2.5)
2 \pscurve[variableLW,startLW=1pt,endLW=20pt,
3   variableColor](-1,0.5)(-2,1)(2,2)(-1,-2)(2,-2)
4 \end{pspicture}

```

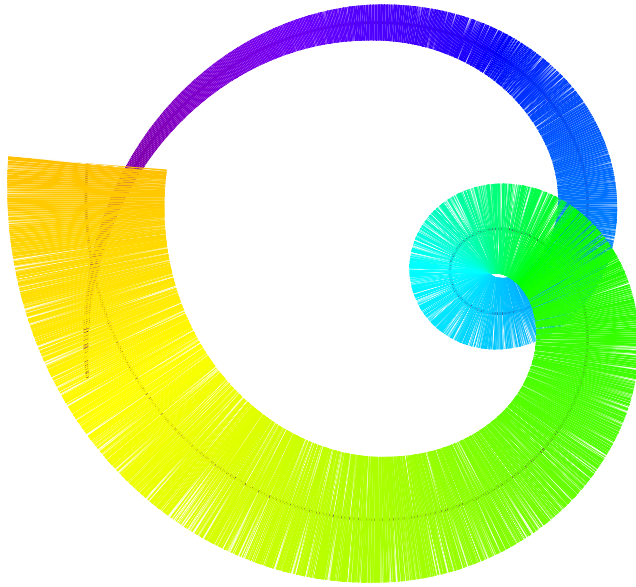
```

1 \begin{pspicture}(-2.5,-2.5)(2.5,2.5)
2 \pscurve[variableLW,startLW=1pt,endLW=20pt]%
3   (-1,0.5)(-2,1)(2,2)(-1,-2)(2,-2)
4 \end{pspicture}

```



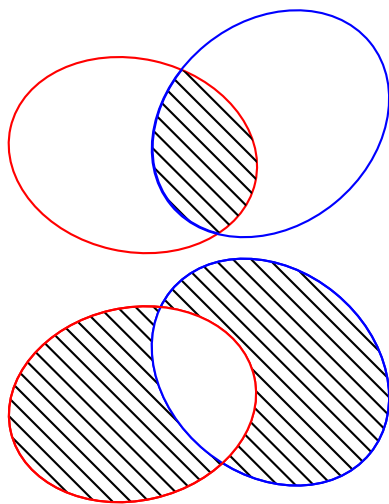
```
1 \begin{pspicture}(-5,-3)(5,3)
2 \psplot[variableLW,startLW=1pt,endLW=20pt,
3   linecolor=magenta!60,variableColor,
4   algebraic,plotpoints=3000,startWL=500,
5   endWL=700]{-5}{5}{2*sin(2*x)+cos(x)}
6 \end{pspicture}
```



```
1 \psset{endLW=24pt}
2 \begin{pspicture}(-5,-5)(5,5)
3 \psparametricplot[variableLW,startLW=1pt,
4   endLW=60pt,linecolor=red,variableColor,
5   algebraic,plotpoints=3000,plotstyle=curve,
6   opacity=0.4,strokeopacity=0.4,
7   endWL=600]{-5}{5}{t*sin(t) | t*cos(t)}
8 \end{pspicture}
```

### 3.3. Macro \psellipse

To rotate an ellipse the already existing keyword `rot` can be used. This is easier than using the `\rput` command and its optional argument for rotating.



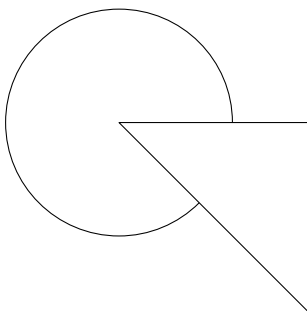
```

1 \psset{unit=0.25}
2 \begin{pspicture}(-1,5)(20,18)
3 \psclip{\psellipse[linecolor=red,
4   rot=-12.606](5.821,10.04)(6.633,5.103)}
5   \psellipse[linecolor=blue,fillstyle=vlines,
6     rot=39.29](13.141,11.721)(6.8,5.4)
7 \endpsclip
8 \psellipse[linecolor=blue,rot=39.29](13.1,11.7)(6.8,5.4)
9 \end{pspicture}
10
11 \begin{pspicture}(-1,5)(20,18)
12 \psellipse[linecolor=blue,rot=-39.29,
13   fillstyle=vlines](13.1,11.7)(6.8,5.4)
14 \psclip{\psellipse[linecolor=red,rot=12.6,
15   fillstyle=vlines](5.8,10)(6.6,5.1)}
16   \psellipse*[linecolor=white,rot=-39.29](13.1,11.7)
17     (6.8,5.4)
18 \endpsclip
19 \psellipse[linecolor=blue,rot=-39.29](13.1,11.7)
20   (6.8,5.4)
21 \psellipse[linecolor=red,rot=12.6](5.8,10)(6.6,5.1)
22 \end{pspicture}

```

### 3.4. Macro \psellipticarc

In a circle the angle is proportional to the bow:  $b = r\alpha$ . In an elliptic arc this is no more the case, which is the reason why angles are internally corrected by PSTricks, to get the same arc lengths for different radii:

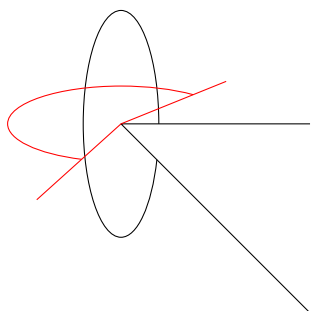


```

1 \psset{unit=0.5cm}
2 \begin{pspicture}(-5.5,-5.5)(5.5,5.5)%
3 \psset{linewidth=0.4pt,linejoin=1}
4 \psline(5,0)(0,0)(5,-5)
5 \psellipticarc(0,0)(3,3){0}{315}
6 \end{pspicture}%

```

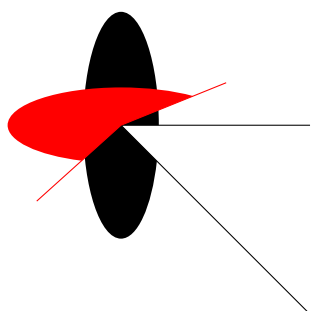




```

1 \psset{unit=0.5cm}
2 \begin{pspicture}(-5.5,-5.5)(5.5,5.5)%
3 \psset{linewidth=0.4pt,linejoin=1}
4 \psline(5,0)(0,0)(5,-5)
5 \psellipticarc(0,0)(1,3){0}{315}%
6 \psset{linecolor=red}
7 \psellipticarc(0,0)(3,1){22}{222}%
8 \psline(3;22)\psline(3;222)
9 \end{pspicture}%

```

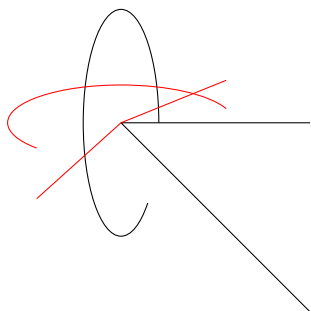


```

1 \psset{unit=0.5cm}
2 \begin{pspicture}(-5.5,-5.5)(5.5,5.5)%
3 \psset{linewidth=0.4pt,linejoin=1}
4 \psline(5,0)(0,0)(5,-5)
5 \psellipticarc*(0,0)(1,3){0}{315}%
6 \psset{linecolor=red}
7 \psellipticarc*(0,0)(3,1){22}{222}%
8 \psline(3;22)\psline(3;222)
9 \end{pspicture}%

```

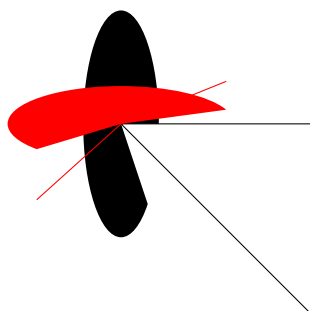
If you do not want the angle correction, then use the keyword setting `correctAngle=false`:



```

1 \psset{unit=0.5cm}
2 \begin{pspicture}(-5.5,-5.5)(5.5,5.5)%
3 \psset{linewidth=0.4pt,linejoin=1,
4 correctAngle=false}
5 \psline(5,0)(0,0)(5,-5)
6 \psellipticarc(0,0)(1,3){0}{315}%
7 \psset{linecolor=red}
8 \psellipticarc(0,0)(3,1){22}{222}%
9 \psline(3;22)\psline(3;222)
10 \end{pspicture}%

```



```

1 \psset{unit=0.5cm}
2 \begin{pspicture}(-5.5,-5.5)(5.5,5.5)%
3 \psset{linewidth=0.4pt,linejoin=1,
4   correctAngle=false}
5 \psline(5,0)(0,0)(5,-5)
6 \psellipticarc*(0,0)(1,3){0}{315}%
7 \psset{linecolor=red}
8 \psellipticarc*(0,0)(3,1){22}{222}%
9 \psline(3;22)\psline(3;222)
10 \end{pspicture}%

```

### 3.5. Option algebraic

The option algebraic moved from the other packages into the main package pstricks to get rid of the dependencies.

By default the function in \psplot has to be described in Reversed Polish Notation. The option algebraic allows you to do this in the common algebraic notation. E.g.:

RPN	algebraic
x ln	ln(x)
x cos 2.71 x neg 10 div exp mul	cos(x)*2.71^(-x/10)
1 x div cos 4 mul	4*cos(1/x)
t cos t sin	cos(t) sin(t)

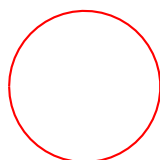
Setting the option algebraic, allow the user to describe all expression to be written in the classical algebraic notation (infix notation). The four arithmetic operations are obviously defined +-\*/, and also the exponential operator ^. The natural priorities are used :  $3 + 4 \times 5^5 = 3 + (4 \times (5^5))$ , and by default the computation is done from left to right. The following functions are defined :

sin, cos, tan, acos, asin	in radians
log, ln	
ceiling, floor, truncate, round	
sqrt	square root
abs	absolute value
fact	for the factorial
Sum	for building sums
IfTE	for an easy case structure

These options can be used with **all** plot macros.

**Using the option algebraic implies that all angles have to be in radians!**

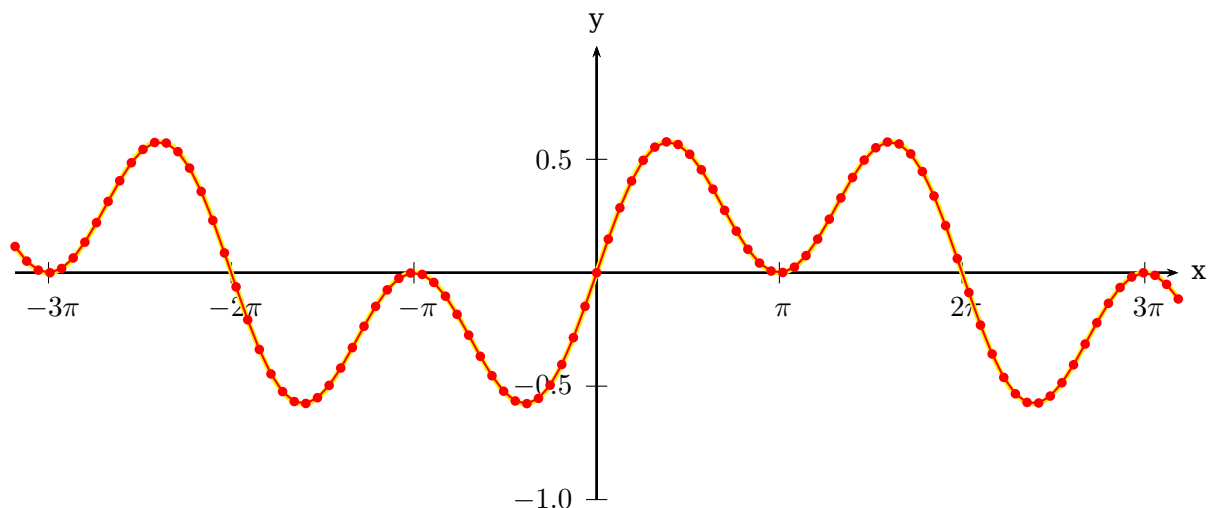
For the \parametricplot the two parts must be divided by the | character:



```

1 \begin{pspicture}(-0.5,-0.5)(0.5,0.5)
2 \parametricplot[algebraic,linecolor=red]{-3.14}{3.14}{cos(t)|sin(t)}
3 \end{pspicture}

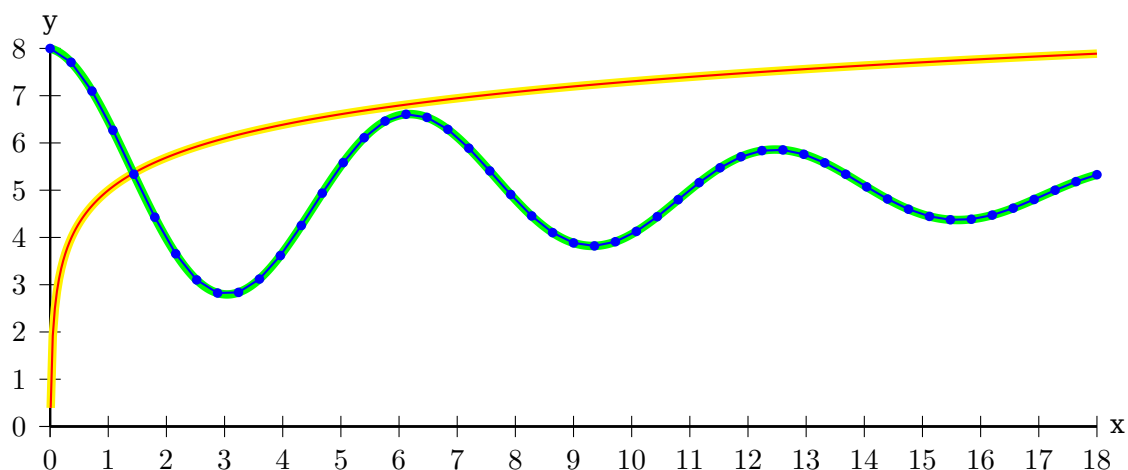
```



```

1 \psset{lly=-0.5cm}
2 \psgraph[trigLabels,dx=\psPi,dy=0.5,Dy=0.5]{->}(0,0)(-10,-1)(10,1){\linewidth
   }{6cm}
3 \psset{algebraic,plotpoints=1000}
4 \psplot[linecolor=yellow,linewidth=2pt]{-10}{10}{0.75*sin(x)*cos(x/2)}
5 \psplot[linecolor=red,showpoints=true,plotpoints=101]{-10}{10}{0.75*sin(x)*
   cos(x/2)}
6 \endpsgraph

```



```

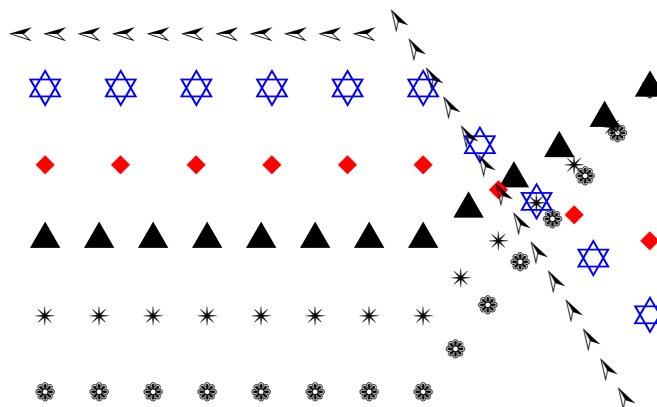
1 \psset{lly=-0.5cm}
2 \psgraph(0,-5)(18,3){0.9\linewidth}{5cm}
3 \psset{algebraic,plotpoints=501}
4 \psplot[linecolor=yellow,linewidth=4\pslinewidth]{0.01}{18}{ln(x)}
5 \psplot[linecolor=red]{0.01}{18}{ln(x)}
6 \psplot[linecolor=yellow,linewidth=4\pslinewidth]{0}{18}{3*cos(x)*2.71^(-x
   /10)}
7 \psplot[linecolor=blue,showpoints=true,plotpoints=51]{0}{18}{3*cos(x)*2.71^(-
   x/10)}
8 \endpsgraph

```

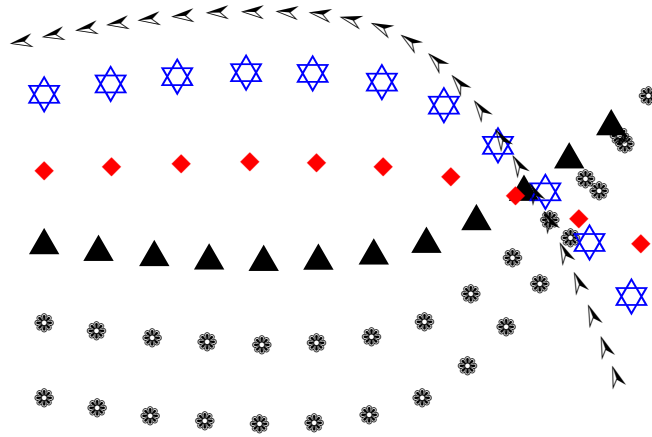
## 4. New linestyle symbol

Instead of drawing a continuous line or curve for a series of coordinates, one can now out a symbol in a given size, direction, and step. This works only for the line style symbol. It takes the symbol defined by the optional argument `symbol`, which can have a single character or a octal number of three digits. The font is specified by the key `symbolFont`, which can take as argument one of the valid PostScript fonts or the internal `PSTricksDotFont`. If the symbol is given by a single character then the equivalent character in the given font is used. The difference between two symbols is set by `symbolStep` and the symbol rotation by `rotateSymbol`. For the first symbol there is an additional keyword `startAngle`. The default values for these new optional keywords are:

```
\psset[pstricks]{symbolStep=20pt}
\psset[pstricks]{symbolWidth=10pt}
\psset[pstricks]{symbolFont=Dingbats}
\psset[pstricks]{rotateSymbol=false}
\psset[pstricks]{startAngle=0}
```



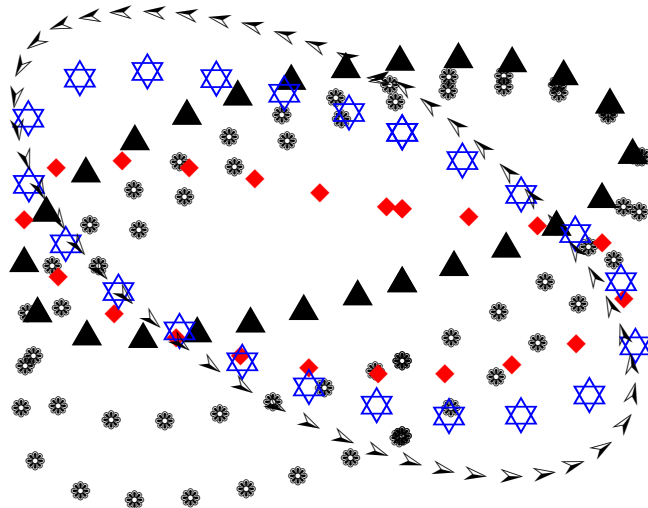
```
1 \pspicture(-1,-1)(8,6)
2 \psline[linestyle=symbol](0,0)(5,0)(8,4)
3 \psline[linestyle=symbol,symbol=T](0,1)(5,1)(8,4)
4 \psline[linestyle=symbol,symbol=u,symbolFont=PSTricksDotFont](0,2)(5,2)(8,4)
5 \psline[linestyle=symbol,symbol=u,symbolStep=25pt,linecolor=red](0,3)(5,3)(8,2)
6 \psline[linestyle=symbol,symbol=A,symbolStep=25pt,
7   symbolWidth=20pt,linecolor=blue](0,4)(5,4)(8,1)
8 \psline[linestyle=symbol,symbol=342,rotateSymbol=true,symbolStep=12pt](0,5)
9   (5,5)(8,0)
10 \endpspicture
```



```

1 \pspicture(-1,-1)(8,6)
2 \pscurve[linestyle=symbol](0,0)(5,0)(8,4)
3 \pscurve[linestyle=symbol](0,1)(5,1)(8,4)
4 \pscurve[linestyle=symbol,symbol=u,symbolFont=PSTricksDotFont](0,2)(5,2)(8,4)
5 \pscurve[linestyle=symbol,symbol=u,symbolStep=25pt,linecolor=red](0,3)(5,3)
   (8,2)
6 \pscurve[linestyle=symbol,symbol=A,symbolStep=25pt,
7   symbolWidth=20pt,linecolor=blue](0,4)(5,4)(8,1)
8 \pscurve[linestyle=symbol,symbol=342,rotateSymbol=true,
9   startAngle=190,symbolStep=12pt](0,5)(5,5)(8,0)
10 \endpspicture

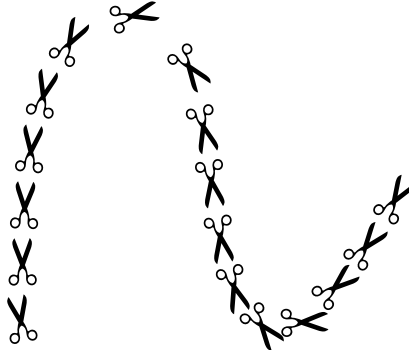
```



```

1 \pspicture(-1,-1)(8,6)
2 \psccurve[linestyle=symbol](0,0)(5,0)(8,4)
3 \psccurve[linestyle=symbol](0,1)(5,1)(8,4)
4 \psccurve[linestyle=symbol,symbol=u,symbolFont=PSTricksDotFont](0,2)(5,2)(8,4)
5 \psccurve[linestyle=symbol,symbol=u,symbolStep=25pt,linecolor=red](0,3)(5,3)
   (8,2)
6 \psccurve[linestyle=symbol,symbol=A,symbolStep=25pt,
7   symbolWidth=20pt,linecolor=blue](0,4)(5,4)(8,1)
8 \psccurve[linestyle=symbol,symbol=342,rotateSymbol=true,
9   startAngle=190,symbolStep=12pt](0,5)(5,5)(8,0)
10 \endpspicture

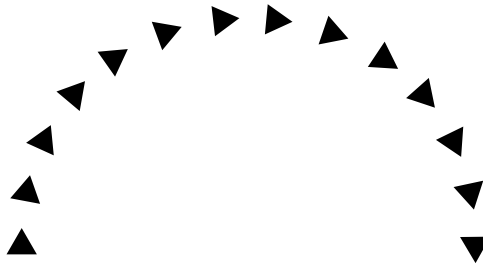
```



```

1 \pspicture(-1,-1)(5,4)
2 \pscurve[rotateSymbol=true,linestyle=symbol,
3   rot=180,startAngle=100,symbol="",
4   symbolWidth=20pt](0,0)(1,4)(3,0)(5,2)
5 \endpspicture

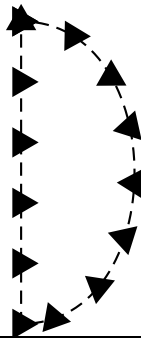
```



```

1 \pspicture(-1,-1)(6,4)
2 \psbezier[rotateSymbol=true,linestyle=symbol,symbol=u,
3 symbolFont=PSTricksDotFont,rot=-90,startAngle=0](0,0)(0,4)(6,4)(6,0)
4 \endpspicture

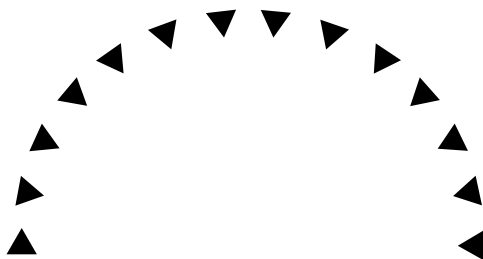
```



```

1 \psset{unit=0.5cm}
2 \pspicture(-1,-4)(6,4)
3 \pscbzier[rotateSymbol=true,linestyle=symbol,symbol=u,
4 symbolFont=PSTricksDotFont](0,4)(4,4)(4,-4)(0,-4)
5 \pscbzier[linestyle=dashed](0,4)(4,4)(4,-4)(0,-4)
6 \endpspicture

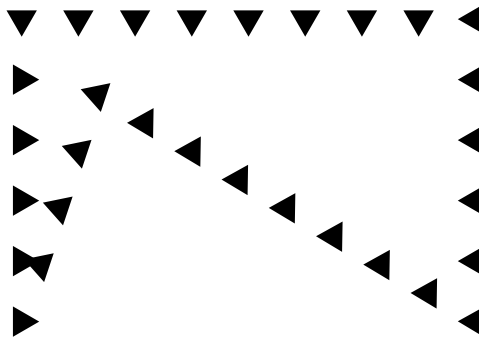
```



```

1 \pspicture(-1,-1)(6,4)
2 \psbezier[rotateSymbol=true,linestyle=symbol,symbol=u,
3 symbolFont=PSTricksDotFont](0,0)(0,4)(6,4)(6,0)
4 \endpspicture

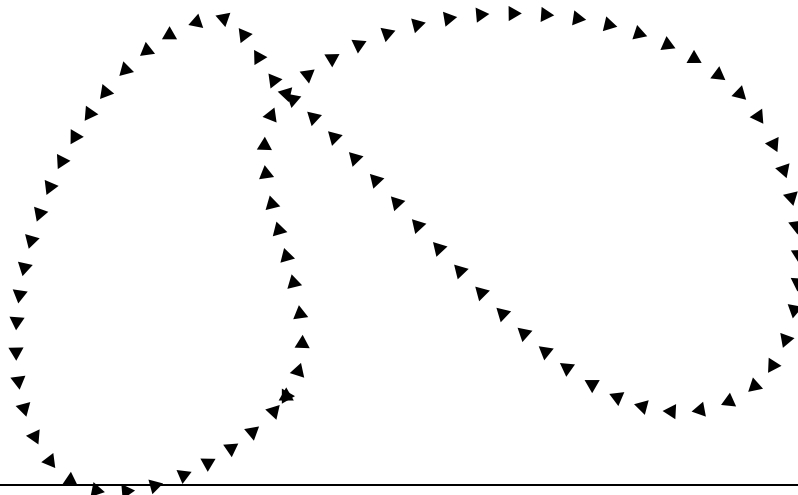
```



```

1 \pspicture(-1,-1)(6,4)
2 \pspolygon[rotateSymbol=true,linestyle=symbol,symbol=u,
3  symbolFont=PSTricksDotFont](0,0)(0,4)(6,4)(6,0)(1,3)
4 \endpspicture

```

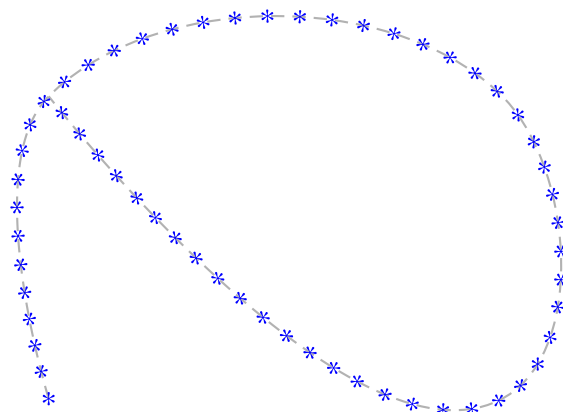


```

1 \pspicture(-3,-1)(6,6)
2 \psccurve[linestyle=symbol,symbol=u, rot=-90,rotateSymbol,
3  symbolFont=PSTricksDotFont, symbolWidth=5pt, symbolStep=10pt
4 ](-3,-1)(0,0)(0,4)(6,4)(6,0)(0,4)(-1,5)
5 \endpspicture

```





```

1 \pspicture(-1,-1)(6,6)
2 \pscurve[linestyle=dashed,linecolor=black!30](0,0)(0,4)(6,4)(6,0)(0,4)
3 \pscurve[rotateSymbol=true,linestyle=symbol,symbol=k,
4   symbolFont=PSTricksDotFont,symbolWidth=5pt,symbolStep=10pt,linecolor=blue
5 ](0,0)(0,4)(6,4)(6,0)(0,4)
6 \endpspicture

```

## 5. Numeric functions

All macros have a @ in their name, because they are only for internal use, but it is no problem to use them like other macros. One can define another name without a @:

```

\makeatletter
\let\pstdivide\pst@divide
\makeatother

```

or put the macro inside the \makeatletter – \makeatother sequence.

## 6. Numeric functions

By default PSTricks loads the file pst-fp which is derived from the fp package. It supports the following macros:

### 6.1. \pstFPadd, \pstFPsub, \pstFPMul, and \pstFPdiv

Multiplication and division:

```

\pstFPadd{result}{number}{number}
\pstFPsub{result}{number}{number}
\pstFPMul{result}{number}{number}
\pstFPdiv{result}{number}{number}

```

-0.079847250000000000	1 \pstFPMul\Result{-3.405}{0.02345} \Result\quad
-145.202558635394456289	2 \pstFPdiv\Result{-3.405}{0.02345} \Result\\
-0.079847250000000000	3 \pstFPMul\Result{0.02345}{-3.405} \Result\quad
-0.006886930983847283	4 \pstFPdiv\Result{0.02345}{-3.405} \Result\\
7726.059000000000000000	5 \pstFPMul\Result{234.123}{33} \Result\quad
7.0946363636363636	6 \pstFPdiv\Result{234.123}{33} \Result\\
267.123000000000000000	7 \pstFPadd\Result{234.123}{33} \Result\quad
201.123000000000000000	8 \pstFPadd\Result{234.123}{-33} \Result\\
201.123000000000000000	9 \pstFPsub\Result{234.123}{33} \Result\quad
-267.123000000000000000	10 \pstFPsub\Result{-234.123}{33} \Result

The zeros can be stripped with the macro \pstFPstripZeros. Expect always rounding errors,  $\TeX$  was not made for calculations ... The value is converted into a length and then reconverted to a number by stripping the unit. Which also strips the zeros.

	1 \pstFPMul\Result{-3.405}{0.02345}
	2 \pstFPstripZeros{\Result}{\Result}\Result\quad
	3 \pstFPdiv\Result{-3.405}{0.02345}
	4 \pstFPstripZeros{\Result}{\Result}\Result\\
-0.07985    -145.20256	5 \pstFPMul\Result{0.02345}{-3.405}
-0.07985    -0.00688	6 \pstFPstripZeros{\Result}{\Result}\Result\quad
	7 \pstFPdiv\Result{0.02345}{-3.405}
	8 \pstFPstripZeros{\Result}{\Result}\Result

## 6.2. \pstFPMul and \pstFPDiv

Integer multiplication and division:

<p>\pstFPMul{result as a truncated integer}{number}{number}</p> <p>\pstFPDiv{result as a truncated integer}{number}{number}</p>
---

-0    -145	1 \makeatletter
-79   -6	2 \pstFPMul\Result{-34.05}{0.02345} \Result\quad
7726   7	3 \pstFPdiv\Result{-3.405}{0.02345} \Result\\
	4 \pstFPMul\Result{23.45}{-3.405} \Result\quad
	5 \pstFPdiv\Result{0.2345}{-0.03405} \Result\\
	6 \pstFPMul\Result{234.123}{33} \Result\quad
	7 \pstFPdiv\Result{234.123}{33} \Result
	8 \makeatother

## 7. The PostScript header files

### 7.1. pstricks.pro

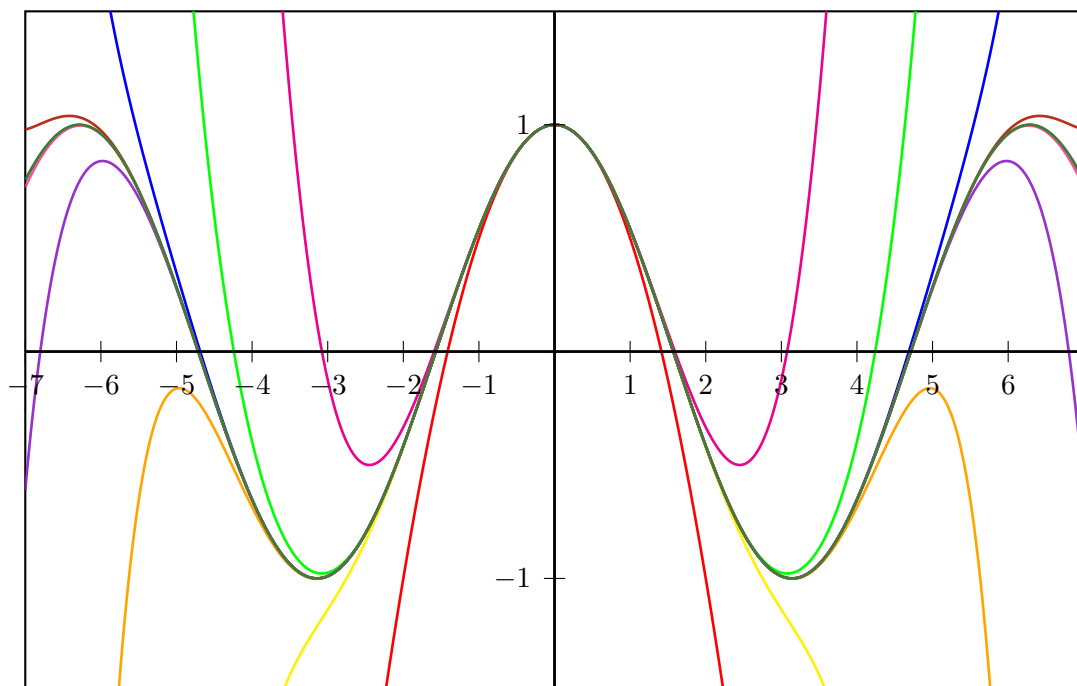
It contains now most of the stuff from pstricks-add and the new routines for plotting lines/curves with symbols.

## 7.2. pst-algparser.pro

### Using the Sum function

`\Sum(<index name>,<start>,<step>,<end>,<function>)`

Let's plot the first development of cosine with polynomials:  $\sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n}}{n!}$ .



```

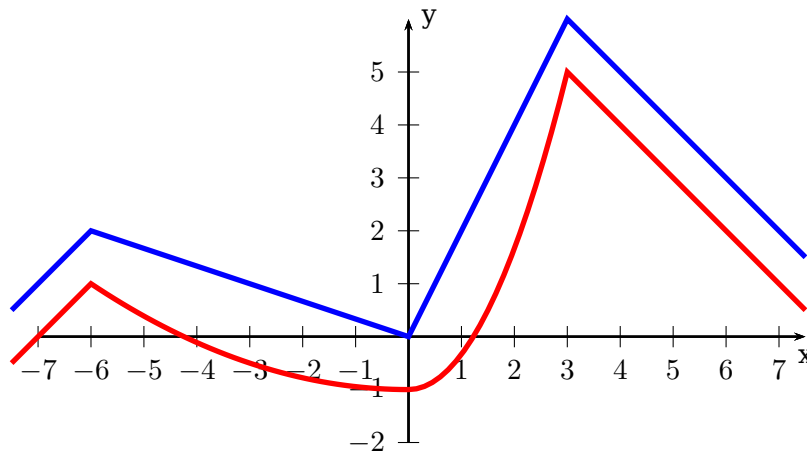
1 \psset{algebraic=true, plotpoints=501, yunit=3}
2 \def\getColor#1{\ifcase#1 black\or red\or magenta\or yellow\or green\or Orange\
  or blue\or
3   DarkOrchid\or BrickRed\or Rhodamine\or OliveGreen\fi}
4 \begin{pspicture}(-7,-1.5)(7,1.5)
5   \psclip{\psframe(-7,-1.5)(7,1.5)}
6   \psplot{-7}{7}{cos(x)}
7   \multido{\n=1+1}{10}{%
8     \psplot[linewidth=1pt,linecolor=\getColor{\n}]{-7}{7}{%
9       Sum(ijk,0,1,\n,(-1)^ijk*x^(2*ijk)/fact(2*ijk))}}
10  \endpsclip
11  \psaxes(0,0)(-7,-1.5)(7,1.5)
12 \end{pspicture}

```

### 7.3. The variable step algorithm together with the PostScript function IfTE

$\text{IfTE}(\langle \text{condition} \rangle, \langle \text{true part} \rangle, \langle \text{false part} \rangle)$

Nesting of several IfTE is possible and seen in the following examples. A classic example is a piece-wise linear function.

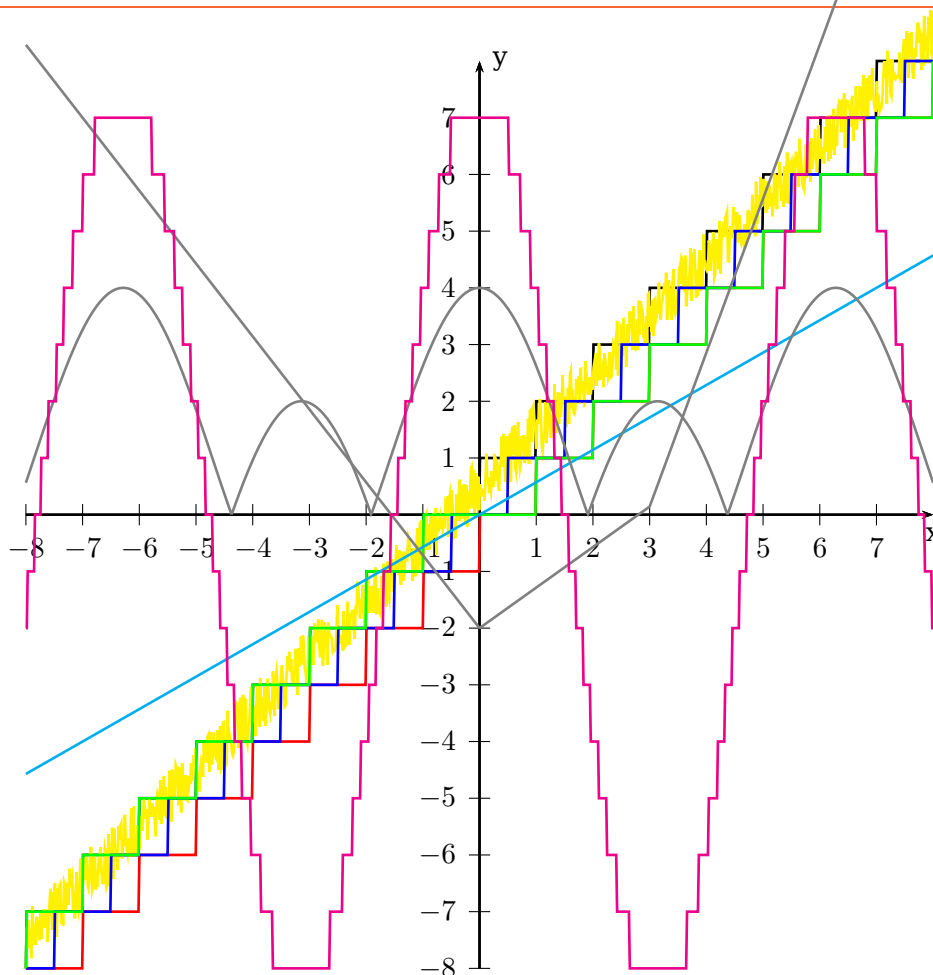


```

1 \psset{unit=1.5, algebraic, VarStep, showpoints, VarStepEpsilon=.001}
2 \begin{pspicture}[showgrid=true](-7,-2)(2,4)
3   \psplot{-7}{2}{IfTE(x<-5, -(x+5)^3/2, IfTE(x<0, 0, x^2))}
4   \psplot{-7}{2}{5*x/9+26/9}
5   \psplot[linecolor=blue]{-7}{2}{(x+7)^30/9^30*4.5-1/2}
6   \psplot[linecolor=red]{-6.9}{2}
7     {IfTE(x<-6, ln(x+7), IfTE(x<-3, x+6, IfTE(x<0.1415926, sin(x+3)+3, 3.1415926-x))
8     )}
9 \end{pspicture}

```

When you program a piece-wise defined function you must take care that a plotting point must be put at each point where the description changes. Use `showpoints=true` to see what's going on when there is a problem. You are on the safe side when you choose a big number for `plotpoints`.

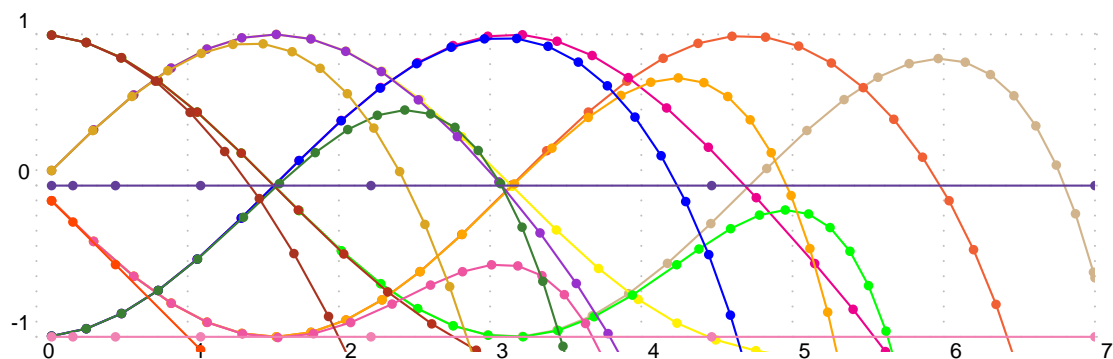


```

1 \psset{unit=0.75}
2 \begin{pspicture}(-8,-8)(8,8)
3   \psaxes{>->}(0,0)(-8,-8)(8,8)[x,-90][y,0]
4   \psset{plotpoints=1000,linewidth=1pt}
5   \psplot[algebraic=true, linecolor=yellow]{-8}{8}{rand/(2^31-1)+x}
6   \psplot[algebraic=true]{-8}{8}{ceiling(x)}
7   \psplot[algebraic=true, linecolor=red]{-8}{8}{floor(x)}
8   \psplot[algebraic=true, linecolor=blue]{-8}{8}{round(x)}
9   \psplot[algebraic=true, linecolor=green]{-8}{8}{truncate(x)}
10  \psplot[algebraic=true, linecolor=cyan]{-8}{8}{div(mul(4,x),7)}
11  \psplot[algebraic=true, linecolor=gray]{-8}{8}{abs(x)+abs(x-3)-abs(5-5*x/7)}
12  \psplot[algebraic=true, linecolor=gray]{-8}{8}{abs(3*cos(x)+1)}
13  \psplot[algebraic=true, linecolor=magenta]{-8}{8}{floor(8*cos(x))}
14 \end{pspicture}

```

## 7.4. Successive derivatives of a polynomial with the PostScript function Derive



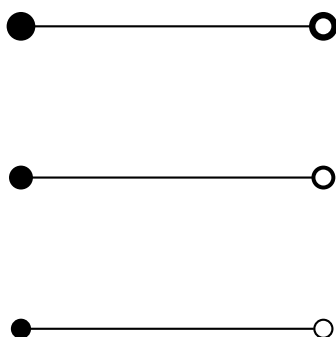
```

1 \psset{unit=2, algebraic=true, VarStep=true, showpoints=true, VarStepEpsilon
   =.001}
2 \def\getColor#1{\ifcase#1 Tan\or RedOrange\or magenta\or yellow\or green\or
   Orange\or blue\or
3   DarkOrchid\or BrickRed\or Rhodamine\or OliveGreen\or Goldenrod\or Mahogany\or
4   OrangeRed\or CarnationPink\or RoyalPurple\or Lavender\fi}
5 \begin{pspicture}[showgrid=true](0,-1.2)(7,1.5)
6   \psclip{\psframe[linestyle=none](0,-1.1)(7,1.1)}
7   \multido{\in=0+1}{16}{%
8     \psplot[algebraic=true, linecolor=\getColor{\in}]{0.1}{7}
9     {Derive(\in,Sum(i,0,1,7,(-1)^i*x^(2*i)/Fact(2*i)))}}
10  \endpsclip
11 \end{pspicture}

```

## 7.5. Special arrow option arrowLW

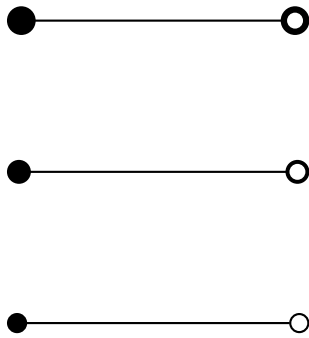
Only for the arrowtype 0, oo, \*, and \*\* it is possible to set the arrowlinewidth with the optional keyword arrowLW. When scaling an arrow by the keyword arrowScale the width of the borderline is also scaled. With the optional argument arrowLW the line width can be set separately and is not taken into account by the scaling value.



```

1 \begin{pspicture}(4,6)
2   \psline[arrowscale=3,arrows=-o](0,5)(4,5)
3   \psline[arrowscale=3,arrows=-o,
4     arrowLW=0.5pt](0,3)(4,3)
5   \psline[arrowscale=3,arrows=-o,
6     arrowLW=0.3333\pslinewidth](0,1)(4,1)
7 \end{pspicture}

```



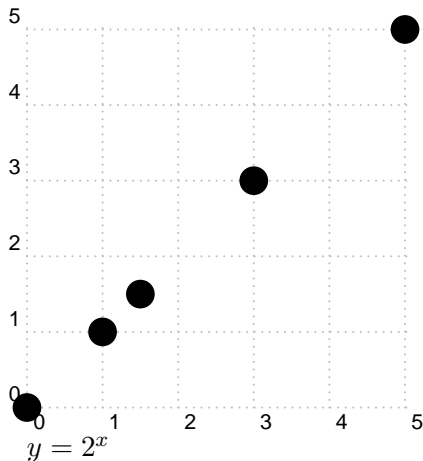
```
1 \begin{pspicture}(4,6)
2 \psline[arrowscale=3,arrows=**-oo](0,5)(4,5)
3 \psline[arrowscale=3,arrows=**-oo,
4   arrowLW=0.5pt](0,3)(4,3)
5 \psline[arrowscale=3,arrows=**-oo,
6   arrowLW=0.3333\pslinewidth](0,1)(4,1)
7 \end{pspicture}
```

## 8. \psforeach and \psForeach

The macro \psforeach allows a loop with an individual increment.

```
\psforeach{variable}{value list}{action}
\psForeach{variable}{value list}{action}
```

With \psforeach the *action* is done inside a group and for \psForeach not. This maybe useful when using the macro to create tabular cells, which are already grouped itself.



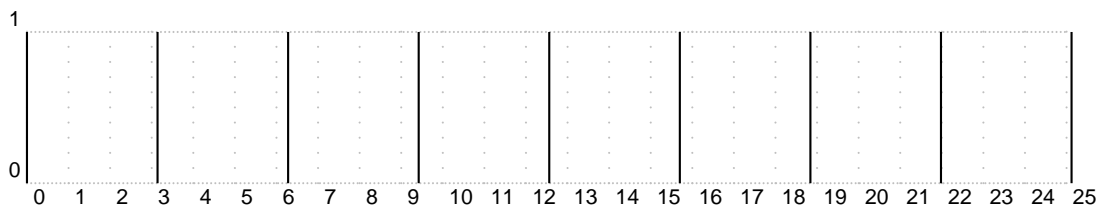
```
1 \begin{pspicture}[showgrid=true](5,5)
2   \psforeach{\nA}{0, 1, 1.5, 3, 5}{%
3     \psdot[dotsize=3](\nA,\nA)}
4 \end{pspicture}
```

2	4	6	8	10	12
4.0	16.0	64.0	256.0	1024.0	4096.0

```
1 %\usepackage{pst-func}
2 \makeatletter
3 \newcommand*\InitToks{\toks@={}}
4 \newcommand\AddToks[1]{\toks@=\expandafter{\the\toks@ #1}}
5 \newcommand*\PrintToks{\the\toks@}
6 \newcommand*\makeTable}[4][5mm]{%
7   \begin{group}
8     \InitToks%
9     \AddToks{\begin{tabular}}{|*{#2}{>{\RaggedLeft}p{#1}}|@{}l@{}}\cline{1-#2}}
10    \psForeach{\iA}{#3}{\expandafter\AddToks\expandafter{\iA & }}
11    \AddToks{\\ \cline{1-#2}}%
12    \psForeach{\iA}{#3}{\expandafter\AddToks\expandafter{\expandafter%
13      \psPrintValue\expandafter{\iA\space /x ED #4} & }}
14    \AddToks{\\ \cline{1-#2}\end{tabular}}%
15    \PrintToks
16  \end{group}
17 }
18 \makeatother
19
20 \sffamily
21 \psset{decimals=2,valuewidth=7,xShift=-20}
22 $y=2^x$\\
23 \makeTable[1cm]{6}{2,4,6,8,10,12}{2 x exp}
```

The value List can also be given by the first two and the last value, e.g. 1,4,...,31, then PSTricks calculates all values with the distance given by the first two values.



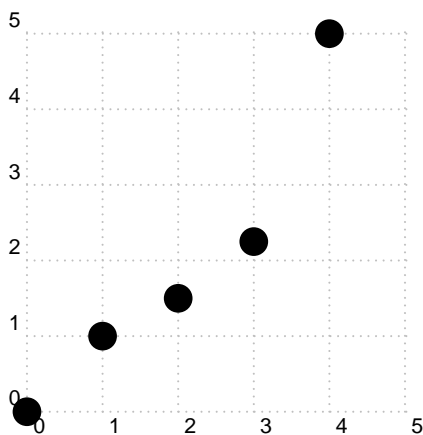


```

1 \psset{xunit=0.55cm,yunit=2cm}
2 \begin{pspicture}[showgrid](0,-5mm)(25,1)
3   \psforeach{\nA}{0, 3.14,...,25}{\psline(\nA,0)(\nA,1)}
4 \end{pspicture}

```

The internal counter for the steps is named `psLoopIndex` and can be used for own purposes.



```

1 \begin{pspicture}[showgrid=true](5,5)
2 \psforeach{\nA}{0, 1, 1.5, 2.25, 5}{%
3   \psdot[dotscale=3](\the\psLoopIndex,\nA)}
4 \end{pspicture}

```

## Part II.

# pst-node – package

## 9. pst-node.tex

The package `pst-node` now uses the advanced key handling from `xkeyval`. The reason why it moved from the base into the contrib sections, where all packages uses `xkeyval`.

## Part III.

# pst-plot – package

## 10. pst-plot.tex

The package `pst-plot` now uses the advanced key handling from `xkeyval`. The reason why it moved from the base into the contrib sections, where all packages uses `xkeyval`.

## 11. List of all optional arguments for pstricks

Key	Type	Default
-----	------	---------

---

### References

- [1] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voß. *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion*. Addison-Wesley Publishing Company, Reading, Mass., 2007.
- [2] Laura E. Jackson and Herbert Voß. Die Plot-Funktionen von pst-plot. *Die T<sub>E</sub>Xnische Komödie*, 2/02:27–34, June 2002.
- [3] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.
- [4] Herbert Voß. Die mathematischen Funktionen von Postscript. *Die T<sub>E</sub>Xnische Komödie*, 1/02:40–47, March 2002.
- [5] Herbert Voss. *PSTricks Support for pdf*. <http://PSTricks.de/pdf/pdfoutput.phtml>, 2002.
- [6] Herbert Voß. *L<sup>A</sup>T<sub>E</sub>X Referenz*. DANTE – Lehmanns, Heidelberg/Hamburg, 1. edition, 2007.
- [7] Herbert Voß. *PSTricks – Grafik für T<sub>E</sub>X und L<sup>A</sup>T<sub>E</sub>X*. DANTE – Lehmanns Media, Heidelberg/Hamburg, 5. edition, 2008.
- [8] Michael Wiedmann and Peter Karp. *References for T<sub>E</sub>X and Friends*. <http://www.miwie.org/tex-refs/>, 2003.

## Index

- [\\*, 22](#)
- [\\*\\*, 22](#)
- [\AddToShipoutPicture, 3](#)
- [algebraic, 10](#)
- [arrowLW, 22](#)
- [arrowscale, 22](#)
- [auto-pst-pdf, 3](#)
- Class
  - [pst-doc, 3](#)
  - [scrartcl, 3](#)
- [correctAngle, 9](#)
- [enable-writel8, 3](#)
- [endLW, 4](#)
- [endWL, 4](#)
- Environment
  - [pspicture, 3](#)
- [eso-pic, 3](#)
- File
  - [pst-doc-pdf, 3](#)
  - [pst-fp, 17](#)
- [fp, 17](#)
- [IfTE, 20](#)
- Keyvalue
  - [PSTricksDotFont, 12](#)
  - [symbol, 12](#)
- Keyword
  - [algebraic, 10](#)
  - [arrowLW, 22](#)
  - [arrowscale, 22](#)
  - [correctAngle, 9](#)
  - [endLW, 4](#)
  - [endWL, 4](#)
  - [linecolor, 8](#)
  - [plotpoints, 20](#)
  - [rot, 8](#)
  - [rotateSymbol, 12](#)
  - [showpoints, 20](#)
  - [startAngle, 12](#)
  - [startLW, 4](#)
  - [startWL, 4](#)
  - [symbol, 12](#)
  - [symbolFont, 12](#)
  - [symbolStep, 12](#)
  - [variableColor, 4](#)
  - [variableLW, 4](#)
  - [VarStep, 20](#)
  - [VarStepEpsilon, 20](#)
  - [vlines, 8](#)
- [latex, 3](#)
- [linecolor, 8](#)
- Macro
  - [\AddToShipoutPicture, 3](#)
  - [\makeatletter, 17](#)
  - [\makeatother, 17](#)
  - [\parametricplot, 10](#)
  - [\psarc, 4](#)
  - [\pscircle, 4](#)
  - [\pscurve, 4](#)
  - [\psellipse, 4](#)
  - [\psellipticarc, 4, 8](#)
  - [\psForeach, 24](#)
  - [\psforeach, 24](#)
  - [\pslinewidth, 4](#)
  - [\psparametricplot, 4](#)
  - [\psplot, 4, 10](#)
  - [\pstFPadd, 17](#)
  - [\pstFPdiv, 18](#)
  - [\pstFPdiv, 17](#)
  - [\pstFPMul, 18](#)
  - [\pstFPMul, 17](#)
  - [\pstFPstripZeros, 18](#)
  - [\pstFPsub, 17](#)
  - [\rput, 8](#)
  - [\SpecialCoor, 3](#)
  - [\Sum, 19](#)
  - [\uput, 3](#)
- [\makeatletter, 17](#)
- [\makeatother, 17](#)
- [o, 22](#)
- [oo, 22](#)

Package  
  auto-pst-pdf, 3  
  eso-pic, 3  
  fp, 17  
  pst-node, 25  
  pst-pdf, 3  
  pst-plot, 25  
  pstricks, 10  
  pstricks-add, 18  
  xkeyval, 25  
Package option  
  enable-write18, 3  
  pdf, 3  
  shell-escape, 3  
`\parametricplot`, 10  
pdf, 3  
pdflatex, 3  
plotpoints, 20  
PostScript  
  IfTE, 20  
Program  
  latex, 3  
  pdflatex, 3  
  pst2pdf, 3  
`\psarc`, 4  
`\pscircle`, 4  
`\pscurve`, 4  
`\psellipse`, 4  
`\psellipticarc`, 4, 8  
`\psforeach`, 24  
`\psforeach`, 24  
`\pslinewidth`, 4  
psLoopIndex, 25  
`\psparametricplot`, 4  
pspicture, 3  
`\psplot`, 4, 10  
pst-doc, 3  
pst-doc-pdf, 3  
pst-fp, 17  
pst-node, 25  
pst-pdf, 3  
pst-plot, 25  
pst2pdf, 3  
`\pstFPadd`, 17  
`\pstFPdiv`, 18  
`\pstFPdiv`, 17  
`\pstFPMul`, 18  
`\pstFpmul`, 17  
`\pstFPstripZeros`, 18  
`\pstFPsub`, 17  
pstricks, 10  
pstricks-add, 18  
PSTricksDotFont, 12  
  
rot, 8  
rotateSymbol, 12  
`\rput`, 8  
  
scrartcl, 3  
shell-escape, 3  
showpoints, 20  
`\SpecialCoor`, 3  
startAngle, 12  
startLW, 4  
startWL, 4  
`\Sum`, 19  
symbol, 12  
symbolFont, 12  
symbolStep, 12  
Syntax  
  \*, 22  
  \*\*, 22  
  o, 22  
  oo, 22  
  
`\uput`, 3  
  
variableColor, 4  
variableLW, 4  
VarStep, 20  
VarStepEpsilon, 20  
vlines, 8  
  
xkeyval, 25